How I Built My Portfolio Website

pc2468

1 Introduction

This is the short story of how I took a neat Astro theme and twisted it just enough until it started looking like pc2468.github.io. No heroic thousand-line rewrites, no divine intervention—just picking the right files and poking them until they behaved.

2 Base Repository

I began with: `my-portfolio`.

The repo came with a tidy structure:

- `src/components/` – sections like Hero, Projects, Experience, etc.
- `src/layouts/` – layout wrapper for the site.
- `src/lib/` – data and utilities.
- `src/pages/` – page definitions (mostly `index.astro`).
- `src/styles/` – global CSS.

In other words, most of the work was already done. My part was basically tinkering until the site looked less like a template and more like me.

3 Files I Changed

Here's the "surgical strikes" list—what I changed and why.

3.1 `GlassHeader.tsx`

The stock navbar was fine, but fine is boring. I swapped it for a glass-effect header (inspired by Gothsec/Portfolio) and made a few CSS tweaks in `global.css` to blend it all together. Now it doesn't just sit there—it actually shines.

3.2 `ProjectsSection.tsx`

Tweaked layouts and project rendering. Subtle but important. The idea was to make it look like my projects live there, not like I borrowed them from a stranger's GitHub.

3.3 `HeroSection.tsx`

The old Hero section was basically "Insert Photo Here." I replaced that placeholder with a proper CV preview (embedded PDF + download link). Suddenly, the page stopped looking like a construction site and started looking like a real homepage.

3.4 `ExperienceSection.tsx`

Small layout and text flow fixes. Nothing earth-shattering, just a nudge to keep everything aligned and professional.

3.5 `data.ts`

This one's unavoidable: the file where all personal info hides (name, socials, projects, etc.). I rewrote it top to bottom. That's the only real way the site stops being "someone else's" and becomes yours.

3.6 `index.astro`

This was where I reorganized the show:

- Added `InterestsSection.tsx`.
- Added `EventsSection.tsx`.
- Reincluded `SkillsSection.tsx` with a cleaner layout.

With that, the homepage finally had a rhythm instead of feeling like a random collage.

3.7 `global.css`

Just some polish—color tweaks, transparency, and backgrounds to match the glassy aesthetic. Small changes in CSS go a long way. (The difference between "almost finished" and "actually finished" often lives in five lines of CSS.)

4 New Sections Added

I added three new components under `src/components/`:

- `InterestsSection.tsx` – for my research interests.
- `EventsSection.tsx` – for events and activities.
- `SkillsSection.tsx` – for the skills list.

Nothing fancy, just useful extra pages that made the site more complete.

5 Deployment

Once everything looked decent, it was the usual routine:

```
npm install
npm run build
git add .
git commit -m "deploy"
git push
```

GitHub Pages handled the rest. No late-night debugging marathons, no burnt coffee—just a clean deploy.

6 Conclusion

So that's it. Take a well-made base, edit the parts that matter (`GlassHeader`, `HeroSection`, `data.ts`, `index.astro`, etc.), add a couple of your own sections, fix up the CSS, and deploy.

The result isn't magic—it's just good editing. Like Feynman said about understanding physics: you don't need to know everything, just enough to know where to poke. Turns out, websites aren't that different.